UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/760,429	01/21/2004	Taketo Heishi	2004_0070A	9201

513 7590 08/24/2007

WENDEROTH, LIND & PONACK, L.L.P.
2033 K STREET N. W.
SUITE 800
WASHINGTON, DC 20006-1021

EXAMINER	
CHEN, QING	
ART UNIT	**PAPER NUMBER**
2191	

MAIL DATE	DELIVERY MODE
08/24/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/760,429	HEISHI ET AL.	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 January 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-59* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-59* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 January 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20040414, 20050912, 20060807*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This is the initial Office action based on the application filed on January 21, 2004.

2. **Claims 1-59** are pending.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

4. The abstract of the disclosure is objected to because it exceeds 150 words in length. Correction is required. See MPEP § 608.01(b).

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. **Claims 2-8, 11, 12, 14, 22-26, 28, 29, 32, 34, 39, 40, 43, 44, 47, 49, and 54-56** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 2-4, 22, 28, 40, 43, and 55 recite the limitation "smaller." The term "smaller" is a relative term, which renders the claims indefinite. The term "smaller" is not defined by the claims nor does the specification provide a standard for ascertaining the requisite degree and one

of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claim 23 depends on Claim 22 and, therefore, suffers the same deficiency as Claim 22.

Claims 5-8, 12, 14, 29, 32, 34, 44, 47, and 49 recite the limitation "said cycle." There is insufficient antecedent basis for this limitation in the claims. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "a cycle" for the purpose of further examination.

Claim 11 recites the limitation "close." The term "close" is a relative term, which renders the claim indefinite. The term "close" is not defined by the claim nor does the specification provide a standard for ascertaining the requisite degree and one of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claims 24-26 and 56 recite the limitation "most significantly." The term "most significantly" is a relative term, which renders the claims indefinite. The term "most significantly" is not defined by the claims nor does the specification provide a standard for ascertaining the requisite degree and one of ordinary skill in the art would not be able to

reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claims 39 and 54 recite the limitation "the acceptance unit." There is insufficient antecedent basis for this limitation in the claims. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "an acceptance unit" for the purpose of further examination.

Claim Rejections - 35 USC § 101

7. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

8. **Claims 1-26 and 42-59** are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1-26 are directed to compiler apparatus. However, the recited components of the compiler apparatus appear to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101. Therefore, these claim limitations can be reasonably interpreted as computer program modules—software *per se*. Thus, the claims are directed to functional descriptive material *per se*, and hence non-statutory.

Claims 42-56 are directed to programs—software *per se*. Thus, the claims are directed to functional descriptive material *per se*, and hence non-statutory.

The claims constitute computer programs representing computer listings *per se*. Such descriptions or expressions of the programs are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer program's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element, which defines structural and functional interrelationships between the computer program and the rest of the computer, that permits the computer program's functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035.

Claims 57-59 recite computer-readable recording medium as a claimed element. However, it is noted that the specification describes such computer-readable recording medium as embracing a transmission medium such as the Internet *(see Page 7: 2-5)*. Consequently, the computer-readable recording medium can be reasonably interpreted as carrying electrical signals.

Claims that recite nothing but the physical characteristics of a form of energy, such as a frequency, voltage, or the strength of a magnetic field, define energy or magnetism *per se*, and as such are non-statutory natural phenomena. *O'Reilly v. Morse*, 56 U.S. (15 How.) 62, 112-14 (1853). Moreover, it does not appear that a claim reciting a signal encoded with functional

descriptive material falls within any of the categories of patentable subject matter set forth in § 101.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. **Claims 1-15 and 19-56** are rejected under 35 U.S.C. 102(b) as being anticipated by **Takano et al. (US 5,790,874)**.

As per **Claim 1**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6*

to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");

- an optimization unit operable to optimize the intermediate codes so as to reduce a hamming distance between instructions placed in positions corresponding to the same instruction issue unit in consecutive instruction cycles, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 42-45, "And, sequences of the instructions are modified so as to reduce Hamming distances between bit sequences appearing on the instruction bus without influence to the dependence relation (step 203)."; Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment.")*; and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit optimizes the intermediate codes by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority having a smaller hamming distance from

an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).").*

As per **Claim 3**, the rejection of **Claim 2** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit places an instruction with higher priority in the position corresponding to each of the plurality of instruction issue units, said instruction with higher priority having a smaller hamming distance of an operation code from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).").*

As per **Claim 4**, the rejection of **Claim 2** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit places an instruction with higher priority in the position corresponding to each of the plurality of instruction issue units, said instruction with higher priority having a smaller hamming distance of a register number from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle *(see Column 17: 36-42, "Subsequently, data to be focused in this check is selected (step 505), and then register numbers allocated to focused are replaced with register numbers which enable Hamming distances to be minimized (step 507). At this time, taking Hamming distances between instructions on the boundary of available ranges into consideration, optimization is effected so as to reduce Hamming distances on the boundary.").*

As per **Claim 5**, the rejection of **Claim 1** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit optimizes the intermediate codes by permuting instructions placed in a target instruction cycle within a cycle so as to reduce a hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).").*

As per **Claim 6**, the rejection of **Claim 5** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit permutes the instructions placed in the target instruction cycle within a cycle so as to reduce a sum of hamming distances from a plurality of instructions placed in positions corresponding to the plurality of instruction issue units in the immediately preceding cycle *(see Column 14: 18-21, "In the program before optimization of the basic block B3, for example (refer to FIG. 12), sum total of Hamming distances Hd_sum, Hd_total can be reduced by 10 by such process.").*

As per **Claim 7**, the rejection of **Claim 5** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit permutes the instructions placed in the target instruction cycle within a cycle so as to reduce a hamming distance of an operation code from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).").*

As per **Claim 8**, the rejection of **Claim 5** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit permutes the instructions placed in the target instruction cycle within a cycle so as to reduce a hamming distance of a register number from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle *(see Column 17: 36-42, "Subsequently, data to be focused in this check is selected (step 505), and then register numbers allocated to focused are replaced with register numbers which enable Hamming distances to be minimized (step 507). At this time, taking Hamming distances between instructions on the boundary of available ranges into consideration, optimization is effected so as to reduce Hamming distances on the boundary.").*

As per **Claim 9**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");*

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");*

- an optimization unit operable to optimize the intermediate codes so that a same register is accessed in consecutive instruction cycles, without changing dependency between

instructions corresponding to the intermediate codes *(see Column 16: 33-51, "... add r0, r1, r2 mul r2, r3, r2 ..."; Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment.")*; and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*.

As per **Claim 10**, the rejection of **Claim 9** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit optimizes the intermediate codes by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority being for accessing a register of an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).")*.

As per **Claim 11**, the rejection of **Claim 10** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit places an instruction with higher priority in the position corresponding to each of the plurality of instruction issue units, said instruction with higher priority being for accessing a register with a register number close to a register number of the register of the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).").*

As per **Claim 12**, the rejection of **Claim 9** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit optimizes the intermediate codes by permuting instructions placed in a target instruction cycle within a cycle so that a register of an instruction placed in an immediately preceding cycle is accessed consecutively for a largest number of times, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 17: 36-42, "Subsequently, data to be focused in this check is selected (step 505), and then register numbers allocated to focused are replaced with register numbers which enable Hamming distances to be minimized (step 507). At this time, taking Hamming distances*

between instructions on the boundary of available ranges into consideration, optimization is effected so as to reduce Hamming distances on the boundary.").

As per **Claim 13**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an optimization unit operable to optimize the intermediate codes by placing said predetermined instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between instructions corresponding to the intermediate codes *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205)."; Column 12: 28-30, "The assembly source program*

shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment."); and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.").*

As per **Claim 14**, the rejection of **Claim 13** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit optimizes the intermediate codes by permuting instructions placed in a target instruction cycle within a cycle so that the permuted instructions match a largest number of instructions in a set of instructions including the instruction which is issued with higher priority by each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).").*

As per **Claim 15**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an optimization unit operable to optimize the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units so that no instruction is placed in a position corresponding to a specific instruction issue unit out of said plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205)."; Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment.")*; and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.").*

As per **Claim 19**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");*

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");*

- an optimization unit operable to optimize the intermediate codes by placing instructions so as to operate only a specified number of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 42-45, "And, sequences of the instructions are modified so as to reduce Hamming distances between bit sequences appearing on the instruction bus without influence to the dependence relation (step*

203)."; Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment."); and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*.

As per **Claim 20**, the rejection of **Claim 19** is incorporated; and Takano et al. further disclose:

- wherein the source program includes unit number specification information specifying the number of instruction issue units used by the processor *(see Column 12: 30-33, "In this assembly source program, the basic blocks are retrieved in step 301 in FIG. 3. Thereby, this program is divided into six basic blocks as shown by B1 to B6 in FIGS. 6 to 9.")*, and

- the optimization unit optimizes the intermediate codes by placing the instructions so as to operate only the instruction issue units of the number specified by the unit number specification information, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 12: 52-54, "First, optimization process of the first basic block B1 will be explained. The basic block B1 shows prologue process for preserving inner states to execute main program.")*.

As per **Claim 21**, the rejection of **Claim 19** is incorporated; and Takano et al. further disclose:

- an acceptance unit operable to accept the number of instruction issue units used by the processor *(see Column 12: 30-33, "In this assembly source program, the basic blocks are retrieved in step 301 in FIG. 3. Thereby, this program is divided into six basic blocks as shown by B1 to B6 in FIGS. 6 to 9.")*,

- wherein the optimization unit optimizes the intermediate codes by placing the instructions so as to operate only the instruction issue units of the number accepted by the acceptance unit, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 12: 52-54, "First, optimization process of the first basic block B1 will be explained. The basic block B1 shows prologue process for preserving inner states to execute main program.")*.

As per **Claim 22**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6*

to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");

- an instruction scheduling unit operable to place instructions in positions corresponding to the plurality of instruction issue units without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 42-45, "And, sequences of the instructions are modified so as to reduce Hamming distances between bit sequences appearing on the instruction bus without influence to the dependence relation (step 203).";*
Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment.");

- a register assignment unit operable to assign a register with higher priority to a variable, said register having a register number of a smaller hamming distance from a register number of a register for an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle, and said variable being used by each of the instructions scheduled by the instruction scheduling unit *(see Column 17: 36-42, "Subsequently, data to be focused in this check is selected (step 505), and then register numbers allocated to focused are replaced with register numbers which enable Hamming distances to be minimized (step 507). At this time, taking Hamming distances between instructions on the boundary of available ranges into consideration, optimization is effected so as to reduce Hamming distances on the boundary.")*; and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In*

FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.").

As per **Claim 23**, the rejection of **Claim 22** is incorporated; and Takano et al. further disclose:

- wherein the register assignment unit assigns the register with higher priority to the variable used by each of the scheduled instructions, said register being same as the register for the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle *(see Column 16: 64-67 through Column 17: 1 and 2, "While, if the certain register has its special meaning or its special function as stated above, register numbers cannot be allocated to registers except for the register having the same function as that of the originally allocated register. Therefore, it should be careful since the selection range is restricted.").*

As per **Claim 24**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program*

list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.");

- an optimization unit operable to optimize the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units according to each of a plurality of predetermined placement methods, without changing dependency between the instructions corresponding to the intermediate codes and by selecting one of the placed instructions which is expected to reduce power consumption most significantly during execution of a program from among the plurality of placed instructions *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205)."; Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment.")*; and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*.

As per **Claim 25**, the rejection of **Claim 24** is incorporated; and Takano et al. further disclose:

- an instruction scheduling unit operable to place the instructions in the positions corresponding to the plurality of instruction issue units according to each of the plurality of predetermined placement methods, without changing dependency between the instructions corresponding to the intermediate codes *(see Column 9: 42-45, "And, sequences of the instructions are modified so as to reduce Hamming distances between bit sequences appearing on the instruction bus without influence to the dependence relation (step 203)."; Column 12: 28-30, "The assembly source program shown in FIGS. 6 to 9 are input into the optimization processing apparatus of the first embodiment.")*; and

- a register assignment unit operable to assign registers to variables according to each of a plurality of predetermined assignment methods, said variables being respectively used by the instructions scheduled by the instruction scheduling unit *(see Column 17: 36-42, "Subsequently, data to be focused in this check is selected (step 505), and then register numbers allocated to focused are replaced with register numbers which enable Hamming distances to be minimized (step 507). At this time, taking Hamming distances between instructions on the boundary of available ranges into consideration, optimization is effected so as to reduce Hamming distances on the boundary.")*,

- wherein the plurality of predetermined placement methods used by the instruction scheduling unit and the plurality of predetermined assignment methods used by the register assignment unit are searched using back tracking so as to obtain placement of the instructions which is expected to reduce power consumption most significantly during execution of the

program *(see Column 20: 35-38, "Next, it is determined whether or not respective instructions correspond to instructions to which the third embodiment is applied, i.e., instructions whose replace candidates are registered in the library (step 703).").*

As per **Claim 26**, the rejection of **Claim 25** is incorporated; and Takano et al. further disclose:

- wherein the optimization unit further includes an instruction rescheduling unit operable to permute the instructions in the positions corresponding to the plurality of instruction issue units according to each of a plurality of predetermined permutation methods, said instructions using the variables to which the registers are assigned *(see Column 9: 45-51, "Thus, low power consumption caused on the instruction bus can be achieved by the modification of the instruction sequences. In the step 203, first the instruction sequences are modified and the Hamming distances at that time are determined by a power consumption reducing means (which corresponds to a instruction sequence modifying means) (step 205).")*, and

- the plurality of predetermined placement methods used by the instruction scheduling unit, the plurality of predetermined assignment methods used by the register assignment unit and the plurality of predetermined permutation methods used by the instruction rescheduling unit are searched using back tracking so as to obtain placement of the instructions which is expected to reduce power consumption most significantly during execution of the program *(see Column 20: 35-38, "Next, it is determined whether or not respective instructions correspond to instructions to which the third embodiment is applied, i.e., instructions whose replace candidates are registered in the library (step 703).").*

Claims 27-29 are compilation method claims corresponding to the compiler apparatus claims above (Claims 1, 2, and 5) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1, 2, and 5.

Claims 30-32 are compilation method claims corresponding to the compiler apparatus claims above (Claims 9, 10, and 12) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 9, 10, and 12.

Claims 33 and 34 are compilation method claims corresponding to the compiler apparatus claims above (Claims 13 and 14) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 13 and 14.

Claim 35 is a compilation method claim corresponding to the compiler apparatus claim above (Claim 15) and, therefore, is rejected for the same reason set forth in the rejection of Claim 15.

Claim 36 is a compilation method claim corresponding to the compiler apparatus claim above (Claim 16) and, therefore, is rejected for the same reason set forth in the rejection of Claim 16.

Claims 37-39 are compilation method claims corresponding to the compiler apparatus claims above (Claims 19-21) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 19-21.

Claim 40 is a compilation method claim corresponding to the compiler apparatus claim above (Claim 22) and, therefore, is rejected for the same reason set forth in the rejection of Claim 22.

Claim 41 is a compilation method claim corresponding to the compiler apparatus claim above (Claim 24) and, therefore, is rejected for the same reason set forth in the rejection of Claim 24.

Claims 42-44 are program claims corresponding to the compiler apparatus claims above (Claims 1, 2, and 29) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1, 2, and 29.

Claims 45-47 are program claims corresponding to the compiler apparatus claims above (Claims 9, 10, and 32) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 9, 10, and 32.

Claims 48 and 49 are program claims corresponding to the compiler apparatus claims above (Claims 13 and 14) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 13 and 14.

Claim 50 is a program claim corresponding to the compiler apparatus claim above (Claim 15) and, therefore, is rejected for the same reason set forth in the rejection of Claim 15.

Claim 51 is a program claim corresponding to the compiler apparatus claim above (Claim 16) and, therefore, is rejected for the same reason set forth in the rejection of Claim 16.

Claims 52-54 are program claims corresponding to the compiler apparatus claims above (Claims 19-21) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 19-21.

Claim 55 is a program claim corresponding to the compiler apparatus claim above (Claim 22) and, therefore, is rejected for the same reason set forth in the rejection of Claim 22.

Claim 56 is a program claim corresponding to the compiler apparatus claim above (Claim 24) and, therefore, is rejected for the same reason set forth in the rejection of Claim 24.

11. **Claims 57-59** are rejected under 35 U.S.C. 102(b) as being anticipated by **Mozdzen et al. (US 5,537,656)**.

As per **Claim 57**, Mozdzen et al. disclose:

- wherein the machine language codes include a first instruction to stop an operation of any of the instruction issue units during an interval in which the instructions are not executed for a predetermined number of instruction cycles or more in said any of the instruction issue units *(see Column 4: 57-63, "... the microcode unit 210 also comprises logic for executing a plurality of instructions. The instructions comprise a "shutdown" instruction for stopping the processor 101, transitioning the processor 101 to a reduced power consumption state, and preventing the processor from executing instructions until it is restarted ... ").*

As per **Claim 58**, the rejection of **Claim 57** is incorporated; and Mozdzen et al. further disclose:

- wherein the machine language codes further include a second instruction to resume the operation of said any of the instruction issue units just after the interval has passed *(see Column 4: 57-66, "... the microcode unit 210 also comprises logic for executing a plurality of instructions. The instructions comprise ... a "resume" instruction which transitions the processor*

out of the reduced power consumption state and enables the processor to restart instructions executions.").

As per **Claim 59**, the rejection of **Claim 57** is incorporated; and Mozdzen et al. further disclose:

- wherein the processor includes a program status register that holds values indicating operation conditions of the plurality of instruction issue units *(see Column 4: 37-42, "The instruction unit 206 also comprises an instruction pointer register (not shown) and a prior instruction pointer register (not shown) for holding the current and prior instructions respectively. The instruction pointer controls instruction fetching in the instruction unit 206.")*, and

- the first instruction writes the values indicating the operation conditions of the plurality of instruction issue units into the program status register *(see Column 4: 37-42, "The instruction unit 206 also comprises an instruction pointer register (not shown) and a prior instruction pointer register (not shown) for holding the current and prior instructions respectively. The instruction pointer controls instruction fetching in the instruction unit 206.")*.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. **Claims 16-18** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Takano et al. (US 5,790,874)** in view of **Mozdzen et al. (US 5,537,656)**.

As per **Claim 16**, Takano et al. disclose:

- a parser unit operable to parse the source program *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an intermediate code conversion unit operable to convert the parsed source program into intermediate codes *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*;

- an interval detection unit operable to detect an interval in which no instruction is placed in a predetermined number of positions, out of a plurality of positions corresponding

respectively to the plurality of instruction issue units in which instructions are to be placed, consecutively for a predetermined number of instruction cycles *(see Column 14: 34-40, "Thus, in order to reduce Hamming distances further more, there is a method wherein, for example, the next basic block subsequent to the target basic block and the basic block subsequent to the next basic block etc. are also considered. However, it is desired that, in order to achieve optimization by simple process quickly, processes are divided into basic blocks.")*; and

- a code generation unit operable to convert the optimized intermediate codes into machine language instructions *(see Column 12: 14-19, "FIGS. 6 to 9 show an assembly source program list when the program shown in FIG. 5 is compiled by the "Sun SPARC C compiler". In FIGS. 6 to 9, the first column denotes line numbers, the second column denotes addresses, the third column denotes object codes, and the fourth column denotes assembly sources.")*.

However, Takano et al. do not disclose:

- a first instruction insertion unit operable to insert, into immediately before the interval, an instruction to stop an operation of the instruction issue units corresponding to the positions where no instruction is placed.

Mozdzen et al. disclose:

- a first instruction insertion unit operable to insert, into immediately before the interval, an instruction to stop an operation of the instruction issue units corresponding to the positions where no instruction is placed *(see Column 4: 57-63, "... the microcode unit 210 also comprises logic for executing a plurality of instructions. The instructions comprise a "shutdown" instruction for stopping the processor 101, transitioning the processor 101 to a reduced power*

consumption state, and preventing the processor from executing instructions until it is restarted ... ").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Mozdzen et al. into the teaching of Takano et al. to include a first instruction insertion unit operable to insert, into immediately before the interval, an instruction to stop an operation of the instruction issue units corresponding to the positions where no instruction is placed. The modification would be obvious because one of ordinary skill in the art would be motivated to transition the processor to a reduced power consumption state *(see Mozdzen et al. – Column 4: 57-63).*

As per **Claim 17**, the rejection of **Claim 16** is incorporated; however, Takano et al. do not disclose:

- a second instruction insertion unit operable to insert, into immediately after the interval, an instruction to resume the operation of the instruction issue units corresponding to the positions where no instruction is placed.

Mozdzen et al. disclose:

- a second instruction insertion unit operable to insert, into immediately after the interval, an instruction to resume the operation of the instruction issue units corresponding to the positions where no instruction is placed *(see Column 4: 57-66, "... the microcode unit 210 also comprises logic for executing a plurality of instructions. The instructions comprise ... a "resume" instruction which transitions the processor out of the reduced power consumption state and enables the processor to restart instructions executions.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Mozdzen et al. into the teaching of Takano et al. to include a second instruction insertion unit operable to insert, into immediately after the interval, an instruction to resume the operation of the instruction issue units corresponding to the positions where no instruction is placed. The modification would be obvious because one of ordinary skill in the art would be motivated to transition the processor out of the reduced power consumption state *(see Mozdzen et al. – Column 4: 57-66)*.

As per **Claim 18**, the rejection of **Claim 16** is incorporated; however, Takano et al. do not disclose:

- wherein the processor includes a program status register that holds values indicating operation conditions of the plurality of instruction issue units, and
- the instruction inserted by the first instruction insertion unit writes the values indicating the operation conditions of the plurality of instruction issue units into the program status register.

Mozdzen et al. disclose:

- wherein the processor includes a program status register that holds values indicating operation conditions of the plurality of instruction issue units *(see Column 4: 37-42, "The instruction unit 206 also comprises an instruction pointer register (not shown) and a prior instruction pointer register (not shown) for holding the current and prior instructions respectively. The instruction pointer controls instruction fetching in the instruction unit 206.")*, and

- the instruction inserted by the first instruction insertion unit writes the values indicating the operation conditions of the plurality of instruction issue units into the program status register *(see Column 4: 37-42, "The instruction unit 206 also comprises an instruction pointer register (not shown) and a prior instruction pointer register (not shown) for holding the current and prior instructions respectively. The instruction pointer controls instruction fetching in the instruction unit 206.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Mozdzen et al. into the teaching of Takano et al. to include wherein the processor includes a program status register that holds values indicating operation conditions of the plurality of instruction issue units, and the instruction inserted by the first instruction insertion unit writes the values indicating the operation conditions of the plurality of instruction issue units into the program status register. The modification would be obvious because one of ordinary skill in the art would be motivated to control instruction fetching *(see Mozdzen et al. – Column 4: 37-42)*.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The

Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

WEI ZHEN
SUPERVISORY PATENT EXAMINER

QC / ac
August 15, 2007